Bridging Dolev-Yao Adversaries and Control Systems with Time-Sensitive Channels

Bogdan Groza and Marius Minea

Politehnica University of Timişoara and Institute e-Austria Timişoara [*]
bogdan.groza@aut.upt.ro, marius@cs.upt.ro

Abstract. Defining security objectives for industrial control scenarios is a challenging task due to the subtle interactions between system components and because security goals are often far from obvious. Moreover, there is a persistent gap between formal models for channels and adversaries (usually, transition systems) and models for control systems (differential or recurrent equations). To bind these two realms, we translate control systems into transition systems by means of an abstraction with variable time granularity and compose them with a channel model that is controlled by Dolev-Yao adversaries. This opens the road for automatic reasoning about the formal model of a control system using model checkers in a context where the communication channel is tampered with. We address a security objective that has so far largely eluded in models, namely freshness, which is highly relevant for control systems. Beyond the traditional resilience to replay attacks, we point out several flavours of freshness which are often overlooked, e.g., ordering and bounded lifespan. We formalize these notions and show that their absence can lead to attacks that subvert the control system. Finally, we build a proof-of-concept implementation that we use to determine attacks on a simple model which clearly shows that real-world scenarios are within reach.

Keywords: control system, formal modelling, freshness

1 Introduction

CONTEXT AND REALISM. The generic image of a control system is that of a closed loop in which a controller regulates the behaviour of a process (usually called plant) as shown in Figure 1. The design of such systems is commonly based on intricacies known only to producers and insiders that use them. However, an important aspect reconfirmed time and again by incidents such as Stuxnet that is that the internal details of a system are hard to be kept secret to well motivated outsiders (likely, the worm exploited specific system details). Relying on security through obscurity is a fatal flaw. Since for cryptographic protocols, modeling has proved to be a crucial tool to assess security, it is quite obvious that modeling industrial control systems in their relation to the communication channels and adversaries is much more relevant today as they become exposed as parts of Internet-like structures, e.g., Internet of Things (IoT) [1]. Indeed, there has been constant attention in the previous years on attack surfaces and countermeasures for control

[*] This work is supported in part by FP7-ICT-2009-5 project 257876, SPaCIoS: Secure Provision and Consumption in the Internet of Services

systems [3, 14], but these lines of work are not based on Dolev-Yao adversary models. While several related cases of modeling industrial communication channels exist, for example Modbus [6] or Fieldbus [4], (the former using tools from the AVISPA project, a precursory of the tools that we employ here) we are unaware of any related work that tries to combine the behaviour of the control system with the communication channel and the adversary abilities.

FRESHNESS. The sparsity of work combining control systems and formal adversary models is compounded by the marginal interest toward certain objectives such as freshness. So far, freshness in security protocols has been interpreted mostly in the traditional sense that disallows the same message to be accepted twice by a principal, e.g., prevention of replay attacks. Clearly, cryptography provides tools to do more than this: nonces, counters and timestamps, carefully embedded in the protocol can all be used to assure ordering (establishing the order in which messages are issued) or limit lifespan (decide if a particular message is still valid). Syverson [12] presents a taxonomy of replay attacks in which the classical sense is extended to more than the mere replay of a message. The taxonomy considers not only classic replays, but also attacks in which a message (or part of it) is used in a context for which it was not intended, e.g., interleaving, reflection or deflection attacks. Indeed, this taxonomy evades the traditional interpretation of replay attacks as used by BAN logic [2]. However, in general, interleavings, reflections or deflections are treated as violations of non-injective agreement [9] and they are not necessarily related to freshness. Besides Syverson's taxonomy [12] there is little interest to use freshness for more than uniqueness. While all tools employed for automatic formal protocol verification are able to detect replay attacks, to the best of our knowledge there is no support to check for objectives such as ordering or limited lifespan. Efforts to model time-sensitive security goals exist [5] but appear to be rather isolated. While in security protocols there seems to be a limited interest in freshness compared to other security objectives that have been extensively studied (e.g., secrecy, authenticity, etc.), this objective can be critical in control scenarios.

METHODOLOGY AND RESULTS. While disrupting standard security goals such as confidentiality or authenticity is within reach for the adversary modelled here, we are not specifically interested in such goals since there is consistent related work on how to model and assure them. Even if a channel is secure in this sense, freshness is not necessarily guaranteed. Here we address three flavours of freshness: uniqueness, which is the usual meaning, that a message cannot be subject to replay; ordering, if a principal does not accept messages out of order, and bounded lifespan, referring to the delay within which messages are accepted. To model control systems, we translate the state model of a process into a transition system based on a Δ-grain abstraction, resulting in a model that can interact with the adversary of the communication channel. This formal model allows us to assess whether in the presence (or absence) of the previously mentioned freshness flavours an adversary can subvert the control law at will. This opens the road to employ model checkers commonly used to verify security protocols in order to assess the security of a control system in this context. Using a model checker, we analyze the composition of the adversary, the channel model, and the system model to obtain an attack trace. Future work may include testing these attacks on real-world industrial communication channels, here we focus mostly on the theoretical foundations.

2 System Model and Abstraction

2.1 State Space Model

Bridging control engineering and computer science is a challenging task since the former usually models systems by differential (continuous-time) or recurrent (discrete-time) equations while the latter uses finite state automata, transition systems, etc. Still, there is significant interest to bridge the two. The problem of extracting a transition system from a generic discrete (or continuous) time system was studied by Pappas [10]. Further refinements of the methodology can be found in later works by Girard and Pappas [7, 8], where metrics are introduced to characterize the degree to which two systems are (bi)similar. The same framework can be extended to non-linear systems, preoccupation for this can be found in the work of Tazaki and Imura [13].

Definition 1 (Discrete Time System). *A discrete-time system is a tuple of three spaces* $\mathbb{X}, \mathbb{U}, \mathbb{Y}$ *(domain of state, input and output), and two functions* $\mathcal{F}, \mathcal{G}$ *(mapping the current state and input to the next state and output, respectively), i.e.,* $DTS(\mathbb{X}, \mathbb{U}, \mathbb{Y}, \mathcal{F}, \mathcal{G})$: $\begin{cases} x(t+1) = \mathcal{F}(x(t), u(t)) \\ y(t) = \mathcal{G}(x(t), u(t)) \end{cases}$*, where* $x(0) = x_0$ *is some fixed initial state.*

This model is called the state space model since it accounts for the state of the system and is the preferred way to model control systems.

EXAMPLE: A WATER TANK SYSTEM. Consider a mechanical-fluid system where a simple on/off controller tries to preserve a given reference level (r) of fluid in the tank as shown in Figure 1. The controller receives the error $e(t)$ computed as difference between the reference level and the output of the system $y(t)$ (which is the height of fluid in the tank), then outputs the command $u(t)$. A zero-order hold (ZOH) on the feedback loop preserves the value from the current time interval for the next, i.e., the output of the plant at step $t-1$ is the input of the controller at step t. To model the tank we use the equations from [11] which are straightforward from Bernoulli's equations. The physical system parameters are: g gravitational acceleration ($9.8\, m/s^2$), A_1 tank area (m^2), A_2 orifice area (m^2), h_1 height of water in the tank (m), h_2 height of orifice (m) and water inflow $f_1 = 0.004$ (m^3/s). The physical system behaviour is governed by Torricelli's law, that is: $\frac{d}{dt}h_1(t) + A_2\sqrt{2g(h_1(t) - h_2)} = f_1(t)$. This continuous time differential equation can be turned into a discrete time recurrent equation by replacing the differential with a finite difference, e.g., $\frac{d}{dt}h_1(t) = \frac{h_1((n+1)T_s) - h_1(nT_s)}{T_s}$ where T_s is the discretization step. By equating the finite difference with the differential the following discrete input-output model of the system is obtained [11]: $h_1(n) = \frac{1}{A_1}\left[T_s f_1[n-1] + A_1 h_1[n-1] - A_2 T_s\sqrt{2g(h_1(n) - h_2)}\right]$. The discrete input-output model is accurate enough for small values of T_s (in the order of seconds or hundreds of seconds) given that this mechanical-fluid system is not a fast process.

2.2 Δ-grain Abstractions

INTUITION. Clearly, the recurrent equations that define a discrete system can be directly transposed into a transition system. But the state space will likely be very large (even

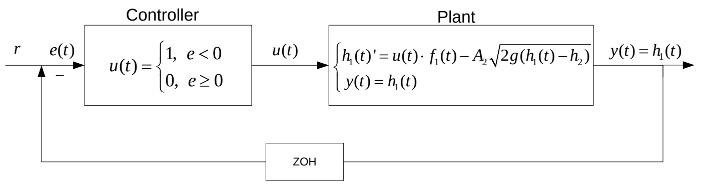
Controller
Plant
r
$e(t)$
$-$
$u(t) = \begin{cases} 1, & e < 0 \\ 0, & e \geq 0 \end{cases}$
$u(t)$
$\begin{cases} h_1(t)' = u(t) \cdot f_1(t) - A_2\sqrt{2g(h_1(t) - h_2)} \\ y(t) = h_1(t) \end{cases}$
$y(t) = h_1(t)$
ZOH

Fig. 1: Flow control system

infinite in case when the state of the system is a real number, e.g., the water level) and the complexity of this model will become prohibitive for our tools. The intention behind the Δ-grain abstraction is to simplify the model of the system in terms of input, output and state spaces as well as transitions between states. This is needed in order to make the model approachable by the formalism of the employed tools, and nevertheless usable (by selecting an abstraction which is just precise enough to check the desired property). The Δ-grain abstraction is flexible in the sense that it doesn't require describing the behaviour of the system at equidistant steps in time. In the Δ-grain abstraction we are merely interested in the fact that the trajectory of the discrete system is eventually (after k steps) in relation with that of the abstraction. Such an abstraction is not unique, for any real system there exist multiple Δ-grain abstractions. Our minimal prescription for designing it, is to select the states of the process that trigger an output change from the controller as well as the states that are the target of the control rule or of the adversary, then to describe the transitions between them. Choosing more abstract states leads to a more accurate model, but which is harder to verify, resulting in a trade-off.

Definition 2 **(Δ-grain abstraction).** *A discrete time system $DTS^\natural(\mathbb{X}^\natural, \mathbb{U}^\natural, \mathbb{Y}^\natural, \mathcal{F}^\natural, \mathcal{G}^\natural)$ is a Δ-grain abstraction of a discrete time system $DTS(\mathbb{X}, \mathbb{U}, \mathbb{Y}, \mathcal{F}, \mathcal{G})$ under relations $\mathcal{R}_x$, $\mathcal{R}_u$, $\mathcal{R}_y$ for states, inputs and outputs iff: i) for any $x \in \mathbb{X}, y \in \mathbb{Y}, u \in \mathbb{U}$ there exist $x^\natural \in \mathbb{X}^\natural, y^\natural \in \mathbb{Y}^\natural, u^\natural \in \mathbb{U}^\natural$ with $(x, x^\natural) \in \mathcal{R}_x$, $(y, y^\natural) \in \mathcal{R}_y$ and $(u, u^\natural) \in \mathcal{R}_u$, ii) for any $x_0 \in \mathbb{X}, u_0 \in \mathbb{U}$, $x_0^\natural \in \mathbb{X}^\natural, u_0^\natural \in \mathbb{U}^\natural$ with $(x_0, x_0^\natural) \in \mathcal{R}_x$, $(u_0, u_0^\natural) \in \mathcal{R}_u$ there exists $0 < k \leq \Delta$ such that $(x(k), x^\natural(1)) \in \mathcal{R}_x$ and $(y(k), y^\natural(1)) \in \mathcal{R}_y$ if the input is constant for k steps, i.e., $u(i) = u_0, i \in [0, k-1]$, and iii) for any $k' < k$, $(x(k'), x^\natural(0)) \in \mathcal{R}_x$ and $(y(k'), y^\natural(0)) \in \mathcal{R}_y$ (all intermediary states and outputs have the same abstraction).*

According to the definition of a discrete time system, for the abstraction we have $x^\natural(1) = \mathcal{F}^\natural(x^\natural(0), u^\natural(0))$ while for the discrete system $x(k) = \mathcal{F}(x(k-1), u(k-1))$ is computed similarly by recurrence from x_0, u_0.

Proposition 1 **(Step-or-hold).** *Let $DTS^\natural$ be a Δ-grain abstraction of a discrete time system DTS and $(x^\natural, u^\natural)$ an abstract state and input pair. For any state and input of the real system DTS that is related to this pair the next state of DTS is uniquely related either to the current state or to the next state of the abstract system, i.e., $\forall x, u, (x, x^\natural) \in \mathcal{R}_x, (u, u^\natural) \in \mathcal{R}_u \Rightarrow (\mathcal{F}(x, u), \chi^\natural) \in \mathcal{R}_x, \chi^\natural \in \{x^\natural, \mathcal{F}^\natural(x^\natural, u^\natural)\}$.*

Proof. Condition ii) of Definition 2 requires that after some number k of steps the state of the real system is related to the state of the abstraction. If $k = 1$ then we have $(\mathcal{F}(x, u), \mathcal{F}^\natural(x^\natural, u^\natural)) \in \mathcal{R}_x$. Otherwise, if $k > 1$ then by condition iii) all intermediary states are part of the same abstraction which means $(\mathcal{F}(x, u), x^\natural) \in \mathcal{R}_x$.

EXAMPLE OF $\mathit{\Delta}$-GRAIN ABSTRACTION. We build a $\mathit{\Delta}$-grain abstraction for the previous water tank model. This model does not account explicitly for a state of the system and we apply the previous definition on its output, i.e., the state plays the role of the output. Assume the controller reference value r is set to 20. Naturally, one abstract state must be associated to the values above the reference and one to the values below it. Let these states be med^- and med^+. These states along with similar states for the adversary target, e.g., $high^-$ and $high^+$, are enough in a minimalistic model. But for better granularity, let us refine the state space of the abstraction (which represents the height of fluid and the output of the system) as $\mathbb{X}^\natural = \mathbb{Y}^\natural = \{vlow^-, vlow^+, low^-, low^+, med^-, med^+, high^-, high^+, vhigh^-, vhigh^+\}$. We set the input space of the abstraction as $\mathbb{U}^\natural = \{\mathit{off}, on\}$ according to the valve close and open commands. We define the relations between the abstract states and the real valued states as shown in Table 1 which summarizes the abstraction so far. Let functions $prec$ and $succ$ return the value that precedes or succeeds another value in the set, e.g., $vlow^+ \leftarrow prec(low^-)$, $high^+ \leftarrow succ(high^-)$, with the extremes as fixed points, $vlow^- \leftarrow prec(vlow^-)$ and $vhigh^+ \leftarrow succ(vhigh^+)$. At each step n we define the output of the plant according to $\mathcal{G}^\natural_{\mathcal{P}}(\mathit{off}, y^\natural(n-1)) = prec(y^\natural(n-1))$ and $\mathcal{G}^\natural_{\mathcal{P}}(on, y^\natural(n-1)) = succ(y^\natural(n-1))$ (this is in fact the value of $y^\natural(n)$). We use the same abstractions as in the case of the plant for the input and output spaces of the controller. Similarly, we define the output of the controller as $\mathcal{G}^\natural_{\mathcal{C}}(u^\natural(n-1)) = \mathit{off}$ iff $lower(u^\natural(n-1), med^-)$ or $u^\natural(n-1) = med^-$ and $\mathcal{G}^\natural_{\mathcal{C}}(u^\natural(n-1)) = on$ iff $higher(u^\natural(n-1), med^-)$. Predicates $lower$ and $higher$ can be inferred from the already defined $prec$ and $succ$. We now show that these form a $\mathit{\Delta}$-grained abstraction (the exact value of $\mathit{\Delta}$ is not relevant for this example as it is merely an upper bound). Relation i) of Definition 2 is clearly satisfied for both the plant and controller: for any state we have defined an abstract state. Relations ii) and iii) are proved as follows. For the plant, in any abstract state the next state will be the predecessor or successor of the current state according to the value of the abstract input: off or on. In the case of the real plant, the input will be either 0 or 1 and given that this input is preserved constant eventually the water level will decrease or increase to a state that is related to the successor of the abstract state that is related to the current state. Since whenever the input of the real plant is preserved constant, the output is a monotonic function, relation iii) is satisfied as well for the plant and until the next abstract state is reached all states are in relation to the current state. For the controller, if the input (which is the water level) is preserved constant, there is no change in the output and the same happens in the real system which proves relation ii). Since the output does not change unless the input changes, relation iii) holds as well for the controller.

Table 1: Abstraction sets and relations with the real system

Abstraction sets	Relations between abstract and real values
$\mathbb{U}^\natural = \{\mathit{off}, on\}$	$\forall x \in [0,5) : (x, vlow^-) \in \mathcal{R}_x, \forall x \in [5,10) : (x, vlow^+) \in \mathcal{R}_x$
$\mathbb{X}^\natural = \mathbb{Y}^\natural = \{vlow^-, vlow^+,$	$\forall x \in [10,15) : (x, low^-) \in \mathcal{R}_x, \forall x \in [15,20) : (x, low^+) \in \mathcal{R}_x$
$low^-, low^+, med^-, med^+,$	$\forall x \in [20,25) : (x, med^-) \in \mathcal{R}_x, \forall x \in [25,30) : (x, med^+) \in \mathcal{R}_x$
$high^-, high^+, vhigh^-, vhigh^+\}$	$\forall x \in [30,35) : (x, high^-) \in \mathcal{R}_x, \forall x \in [35,40) : (x, high^+) \in \mathcal{R}_x$
	$\forall x \in [40,45) : (x, vhigh^+) \in \mathcal{R}_x, \forall x \in [45,50) : (x, vhigh^+) \in \mathcal{R}_x$
	$(0, \mathit{off}) \in \mathcal{R}_u$ $(1, on) \in \mathcal{R}_u$

The next proposition establishes that for any behaviour of the abstraction there exists a behaviour of the real system. This property is important since otherwise an attack over the abstraction may simply be a false-positive alarm.

Proposition 2 (Realizability of the abstract trajectory). *Let $DTS^\natural$ be a Δ-grain abstraction of some real system DTS and consider an initial state x_0 with its corresponding abstraction $x_0^\natural$. For any trajectory of the abstraction $\Sigma^\natural = \{(x^\natural(0), u^\natural(0)), (x^\natural(1), u^\natural(1)), \ldots, (x^\natural(\ell), u^\natural(\ell))\}$ there exists a trajectory of the real system:*

$$\begin{aligned}\Sigma = &\{(x(0), u_0), (x(1), u_0), \ldots, (x(n_1-1), u_0),\\ &(x(n_1), u_1), (x(n_1+1), u_1) \ldots, (x(n_1+n_2-1), u_1),\\ &\ldots,\\ &(x(n_1+n_2+...+n_{\ell-1}), u_{\ell-1})), \ldots, (x(n_1+n_2+...+n_\ell-1), u_{\ell-1}\}\end{aligned}$$

for which it holds that $(x(n_1+n_2+\ldots+n_{i-1}), x^\natural(i)) \in \mathcal{R}_x, \forall i \in \{1..\ell\}$, provided that $(u_i, u^\natural(i)) \in \mathcal{R}_u$, $\forall i \in \{0..\ell-1\}$.

Proof sketch. We prove the statement by induction over ℓ. Let x_0 be the initial state of the system and u_0 the input. Then the abstractions $x_0^\natural$ and $u_0^\natural$ exist by property i) of Definition 2. Thus for $\ell = 0$, the trajectory $\Sigma^\natural = \{(x^\natural(0), u^\natural(0))\}$ exists and there is nothing else to prove. Assume the statement holds for $\ell \geq 0$ and prove it for $\ell+1$. Let $N_\ell = n_1+\ldots+n_{\ell-1}$ and let $\mathcal{F}^n(x)$ be the n-fold application of $\mathcal{F}$ to x, preserving u: $\mathcal{F}^k(x,u) = \mathcal{F}(...(\mathcal{F}(x,u)...), u)$. Since $(u_\ell, u^\natural(\ell)) \in \mathcal{R}_u$ by property ii) of Definition 2 there exists an integer k such that $(\mathcal{F}^k(x(N_\ell), u_\ell), \mathcal{F}^\natural(x^\natural(\ell), u^\natural(\ell))) \in \mathcal{R}_x$, since the input is kept at $u(\ell)$. Choosing $n_l = k$, we clearly have $\mathcal{F}^k(x(N_\ell), u_\ell) = x(N_\ell + n_\ell)$ and thus $(x(n_1+\ldots+n_\ell), x^\natural(\ell+1)) \in \mathcal{R}_x$. Moreover, $\mathcal{F}^j(x(N_\ell), u_\ell) = x(N_\ell)$ for any $j < k$, by property iii), thus the state does not change from $x(N_\ell)$ to $x(N_\ell + n_\ell - 1)$. Thus, the inductive step holds.

While the previous proposition addresses the state trajectory, it trivially extends to the output trajectory since the output directly results from applying $\mathcal{G}$ to state and input.

We next couple an abstraction (playing the role of the plant) with another system (playing the role of a controller). We establish under which circumstances for each trajectory of the coupled abstraction there exists a trajectory of the coupled real system.

Proposition 3. *Let DTS_α and DTS_β be discrete time systems associated to a controller and plant with $DTS_\alpha^\natural$ and $DTS_\beta^\natural$ their Δ-grain abstractions. Let $DTS_{\alpha\rightleftarrows\beta}$ denote the control system that results from the input-output coupling of DTS_α with DTS_β. To avoid a circular dependency, we assume that now the input of DTS_α at step t is the output of DTS_β from step $t-1$ (i.e., they are connected via a zero-order hold on the feedback loop as shown in Figure 1). For any abstract trajectory of $DTS_\beta^\natural$ given as state-input pairs (note that its input is the output of $DTS_\alpha^\natural$):*

$$\Sigma_\beta^\natural = \{(x_\beta^\natural(0), u_\beta^\natural(0)), (x_\beta^\natural(1), u_\beta^\natural(1)), ..., (x_\beta^\natural(n), u_\beta^\natural(n))\}$$

where $\forall i \in [1..n]$, $x_\beta^\natural(i) = \mathcal{F}_\beta^\natural(x_\beta^\natural(i-1), u_\beta^\natural(i-1))$, $u_\beta^\natural(i) = \mathcal{F}_\alpha^\natural(y_\beta^\natural(i), x_\alpha^\natural(i))$, there exists a trajectory of the real system $DTS_{\alpha\rightleftarrows\beta}$ if the following relation exists between *the behaviour of the real and abstract systems: at any step t at which the output of DTS_α changes it holds that i) its input at step t is part of a distinct abstraction than its input at step $t-1$, i.e., $(u_\alpha(t), u^\natural) \in \mathcal{R}_u, (u_\alpha(t-1), v^\natural) \in \mathcal{R}_u, u^\natural \neq v^\natural$, and, due to the delay on the feedback loop ii) the current output of DTS_β and the previous one are part of the same abstraction, i.e., $(y_\beta(t), y^\natural_\beta) \in \mathcal{R}_y$ and $(y_\beta(t-1), y^\natural_\beta) \in \mathcal{R}_y$.*

Proof sketch. We show that for any $t < n$, a trajectory of the real system between t and $t+1$ exists and this is $\varSigma_\beta = \{(x_\beta(t'), u_{\beta,t-1}), (x_\beta(t'+1), u_{\beta,t})$,..., $(x_\beta(t'+k), u_{\beta,t})\}$ where t' denotes the time in the real system at step t in the abstract system. This is explained as follows. The first pair $(x_\beta(t'), u_{\beta,t-1})$ is built with the input from the previous step denoted as $u_{\beta,t-1}$ and this is due to the one-step delay at which the output of DTS_β becomes the input of DTS_α. Condition ii) requires that $(x_\beta(t'), x^\natural_\beta(t)) \in \mathcal{R}_x$ and $(x_\beta(t'+1), x^\natural_\beta(t)) \in \mathcal{R}_x$ (when the input changes the current and previous state are part of the same abstraction). By definition of the $\varDelta$-grained abstraction a k exists such that $(x_\beta(t'+k), x^\natural_\beta(t+1)) \in \mathcal{R}_x$ provided that the input remains constant and this is precisely the case from step $t'+1$ onward since condition i) requires that DTS_α remains unchanged unless its input (which is the output of DTS_β) does not switch to a distinct abstraction. This is assured by condition iii) from the definition of the $\varDelta$-grained abstraction as all intermediary states have the same abstraction.

3 Protocol Model and Adversary Goals

3.1 Protocol Syntax and Properties: Freshness, Ordering and Bounded Lifespan

We assume a standard Dolev-Yao adversary that tampers with the communication channel, including the ability to reorder messages. To express the intruder's ability to delay messages we also need to model time on the communication channel. Consequently we formalize the protocol model as a labeled transition system where each transition corresponds to a tick of a global abstract clock. We begin with a simple framework for protocol execution that allows to define time sensitive properties, then we augment this model with system states. Our framework is different from the usual transitions systems associated to protocol specifications. Usually, a transition system associated to a protocol performs a transition whenever a principal receives a message that fits the expected format and content (as far as the receiver can verify), this is naturally followed by the receiver issuing its response. In our model, a protocol agent may update its state at each step, regardless of a receive action taking place (this is needed as agents will be associated to system abstractions). Moreover, each transition can happen with or without a send event and at each clock tick, several agents may transition. For simplicity, we consider that all agents transition synchronously, possibly without changing their state.

Let $\mathcal{A} = \{A_1, \ldots, A_n\}$ be a set of agent names, $\mathcal{M}$ be a set of messages, and $\mathcal{T} = (\mathcal{A} \times \mathcal{M}) \cup \{\epsilon\}$ be the set of actions, where ϵ is the special empty action.

Definition 3 (Agent model). *A protocol agent (principal) is a state-transition system $\mathcal{P}_i = (\mathcal{S}_i, \mathcal{R}_i, \mathcal{I}_i)$, where $\mathcal{S}_i$ is a set of states, $\mathcal{I}_i \subseteq \mathcal{S}_i$ is a set of initial states, and $\mathcal{R}_i \subseteq \mathcal{S}_i \times \mathcal{T} \times \mathcal{T} \times \mathcal{S}_i$ is the transition relation.*

We also write $(\sigma, r_i, s_i, \sigma') \in \mathcal{R}_i$ as $\sigma \xrightarrow{r_i, s_i} \sigma'$, denoting a transition from state σ to σ', triggered in principal A_i by receive action r_i and executing send action s_i. Each of the two actions is either empty (ϵ) or a pair (A_j, m) representing communication (receive/send) of message $m \in \mathcal{M}$ with principal A_j ($j \neq i$).

For the case of n agents, let $\mathcal{S} = \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$ and $\mathcal{I} = \mathcal{I}_1 \times \ldots \times \mathcal{I}_n$ be the overall and initial state space. For a state $\sigma \in \mathcal{S}$ we denote by σ_i its projection on $\mathcal{S}_i$, likewise, for vectors $\overline{r}, \overline{s} \in \mathcal{T}^n$ we denote by r_i, s_i the individual actions in agent A_i.

Definition 4 (Protocol execution). *A sequence $\rho = \sigma^0 \xrightarrow{\overline{r}^1, \overline{s}^1} \sigma^1 \ldots \xrightarrow{\overline{r}^t, \overline{s}^t} \sigma^t$ is a protocol execution by agents $\mathcal{P}_i = (\mathcal{S}_i, \mathcal{R}_i, \mathcal{I}_i)$ if: i) σ^0 is part of the initial state state space, i.e., $\sigma^0 \in \mathcal{I}$, and ii) the projections of a transition corresponds to a valid transition in each principal, i.e., $\forall j \in 1 \,..\, t, i \in 1 \,..\, n$ it holds $(\sigma_i^{j-1}, r_i^j, s_i^j, \sigma_i^j) \in \mathcal{R}_i$.*

Let $\mathsf{recv}_i(m, j, t)$ be a predicate which is true if A_i receives message m from A_j at time t, and if so, let $\mathsf{sndtime}_j(m)$ be the time at which m was sent by A_j.

Definition 5 (Freshness, ordering and bounded lifespan). *Consider two principals A_{i_0} and A_{i_1} of a protocol specification. We say that for any execution ρ the (bidirectional) communication channel between A_{i_0} and A_{i_1} ensures: i) uniqueness (the traditional sense of freshness), if a message is accepted only once by any principal, i.e., $\mathsf{recv}_{i_b}(m, i_{\neg b}, t_1) \wedge \mathsf{recv}_{i_b}(m, i_{\neg b}, t_2) \Rightarrow t_1 = t_2, \forall b \in \{0, 1\}$, ii) ordering, if in addition to uniqueness, once a message is accepted, only messages sent after it are subsequently accepted, i.e., $\mathsf{recv}_{i_b}(m_1, i_{\neg b}, t_1) \wedge \mathsf{recv}_{i_b}(m_2, i_{\neg b}, t_2) \wedge t_1 < t_2 \Rightarrow \mathsf{sndtime}_{i_{\neg b}}(m_1) < \mathsf{sndtime}_{i_{\neg b}}(m_2), \forall b \in \{0, 1\}$, iii) δ-bounded lifespan, if in addition to uniqueness, messages are accepted no later than a delay δ after the time they were issued, i.e., $\mathsf{recv}_{i_b}(m, i_{\neg b}, t) \Rightarrow t \leq \mathsf{sndtime}_{i_{\neg b}}(m) + \delta, \forall b \in \{0, 1\}$.*

If the channel is not symmetric between A_{i_0} and A_{i_1} and a particular goal is to be assured only from one side to another but not vice-versa, the definition can be easily modified by setting b to either 0 or 1 accordingly.

The case when the sender needs to send a message more than once is addressed in protocol design by adding distinct time variant parameters to each message (to avoid replay), thus two identical messages should never occur (unless this is intended).

3.2 Merging $\varDelta$-grained Abstractions and Adversary Goals

We now simply add system states (via $\varDelta$-grained abstractions) to the protocol execution. Then we formulate the goal of the adversary and give a proposition which establishes under which circumstances the abstract attack trace has a real world instantiation.

Definition 6 (Protocol execution with system states and inputs). *A protocol agent $\mathcal{P}_j = (\mathcal{S}_j, \mathcal{R}_j, \mathcal{I}_j)$ can instantiate the abstraction $DTS_j^\natural$ of a discrete system DTS_j if its state and output evolve according to the $\varDelta$-grain abstraction of DTS_j having as input the message in the receive action and as output the message in the send action. If one or more such agents exist in protocol execution $\rho^\natural = \sigma^0 \xrightarrow{\overline{r}^1, \overline{s}^1} \sigma^1 \ldots \xrightarrow{\overline{r}^t, \overline{s}^t} \sigma^t$ we call this a protocol execution with system states and inputs and we denote by $\mathsf{X}^i = \{x_1^i, .., x_n^i\}$, $\mathsf{U}^i = \{u_1^i, .., u_n^i\}$ the system states and inputs of all agents at step i (for agents that do not instantiate a $DTS^\natural$ all system states x_j^i and inputs u_j^i are empty ϵ).*

The adversary's goal is a set of target states for the plant (which consequently leads to a particular output). In many practical scenarios, including the examples in the following section, the adversary may force the process to go to any value of its choice. Still, this does not mean that the adversary is in control of the system, since the adversary might be able to take the system to a state only for a brief time instance. Our definition of λ-step subversion requires the adversary to gain complete control over the plant for at least λ steps. If subversion can be proved for any value of λ, then it means that the adversary has full control over the system.

Definition 7 (λ-step subversion). *Let the execution with system states and outputs $\rho^\natural = \sigma^0 \xrightarrow{\overline{r}^1,\overline{s}^1} \sigma^1 \dots \xrightarrow{\overline{r}^t,\overline{s}^t} \sigma^t$ of protocol specification $(\mathcal{S},\mathcal{R},\mathcal{I})$ and let $\mathcal{G}_{adv} = \{\mathsf{X}^0_{adv}, \mathsf{X}^1_{adv}, ..., \mathsf{X}^\lambda_{adv}\}$ the goal of the adversary defined over λ transitions. We say that an adversary can perform a λ-step subversion w.r.t. $\mathcal{G}_{adv}$ over the control system if the states in the goal of the adversary hold during all of the last λ steps of the execution, i.e., $\forall i \in [0,\lambda) : X^{t-i} = X^{\lambda-i}_{adv}$.*

To avoid trivial attacks we assume that the adversary goal is distinct from the target of the controller, i.e., the protocol execution in the absence of the adversary.

For concurrent executions of two or more control systems, the abstract attack trace might not have a real world instantiation because one abstract step does not necessarily correspond to the same number of real steps in the systems. Clearly, one way to fix this would be do define an abstraction with the same number of steps, whenever a concurrent execution is needed. To keep the abstractions flexible, in the next proposition we define under what circumstances the attack trace has a real world instantiation.

Proposition 4 (Realizability of the execution). *Let $\rho^\natural = \sigma^0 \xrightarrow{\overline{r}^1,\overline{s}^1} \sigma^1 \dots \xrightarrow{\overline{r}^t,\overline{s}^t} \sigma^t$ a protocol execution with systems states and inputs in the presence of a Dolev-Yao adversary that has access to the input of each plant (i.e., the output of each controller). Denote by $T_p(i)$ the real time step associated to principal p at transition i (due to the Δ-grained abstraction the number of steps in each system is different). Given that the abstract trajectory of each controller-plant ensemble has a real-world counterpart, the execution has a real world instantiation if either: i) the adversary acts independently on each system without using any input from one to build the input for the other, or ii) for any input that is given at step i to the Dolev-Yao adversary, given α, β the minimum integers at which $\mathsf{U}^{i-\alpha} \neq \mathsf{U}^i, \mathsf{U}^{i+\beta} \neq \mathsf{U}^i$ and any two principals p, q it holds $T_p(i) > T_q(i-\alpha)$ and $T_p(i) < T_q(i+\beta)$ (this enforces that at the real step associated to transition i the input is the same in all systems).*

Proof. Case i) is straightforward. If the adversary acts independently on the systems, there always exists a real world instantiation due to Proposition 1. Otherwise, if the adversary subverts inputs from one system to another (i.e., Dolev-Yao behaviour) since the Δ-grained abstraction requires variable number of steps, it may be that at the same step of the abstraction, the number of steps in the real systems is different. But case ii) requires that even if this is the case, the input of the system is unchanged at the time of the subversion, i.e., step i, and therefore whatever the intruder learns at step i of the abstraction, it will be able to learn the same from the real-world instantiation.

In the case of δ-bounded lifespan, one additional issue is to decide whether an input from a previous transition step of the abstraction can still be accepted in the current step. This can be done in two ways. The easiest and more accurate way is to associate each transition of the abstraction with an exact real-world delay (the number of steps needed by the real system). A message issued in the previous abstract time step is accepted only if its lifetime is greater than this value. Otherwise, one could use a globally fixed number of steps for the abstraction (which would be similar to a new discretization), but this will result in a more rigid model that will likely increase the state-space and make the search harder. A rule that can be used is that if $\delta < \Delta$ then a message received by the plant from the controller is either immediately used or will be dropped in the next step (as the maximum possible number of steps in the real system exceeds the message lifespan). If no attack is found, this may be a false negative since Δ is merely a maximum. In contrary, if the message is accepted, the attack may be a false positive. We defer such details for more concrete examples in future work.

3.3 Models and Attacks

We build a minimal, yet meaningful toy model around the fluid-mechanical system from Section 2 and find attack traces on it using a model checker.
TRANSITIONS VS. HORN CLAUSE BASED MODELS. To model the plant and controller, two distinct approaches are possible within the formalism employed by the tools that we work with: transitions and Horn clauses. In the transition based model, a global clock transition allows the plant to move from one state to another and the controller to issue a new command. The state of each principal is updated via a transition as well. In the case of two plants, which makes the attacks more interesting to model, there are four possible transitions for the controller as well as for the plant. The controller can issue for kinds of commands: on for both valves (on, on), on for the first and off for the second (on, off), off for the first and on for the second (off, on) and off for both $(\mathit{off}, \mathit{off})$. The two tanks can receive the correct and most recent command or each of these commands can be deflected by the adversary making the tank preserve the most recent command. Horn clauses allow a more efficient approach as both the controller output and process state can be inferred from atomic Horn clauses at the end of each transition (this significantly reduces the state space). Consequently, the Horn based model can be reduced to a single transition conditioned by the current state of the plant and contoller and the rules (Horn clauses) that define the new state of the controller and plant.
UNIQUENESS AND ORDERING AS A MUST. When uniqueness and ordering are not assured, the attacks are not surprising, λ-step subversion is feasible for any goal. Without uniqueness, once both possible commands are issued, the intruder can reuse them at will to keep the process in the desired state. Without ordering, the intruder can get any number of positive or negative commands (clearly, if the intruder cuts communication to the plant, the controller will continuously issue the same command) and then use them at will to preserve the desired state. Finding attack traces for this case is easy and for brevity we focus on finding attacks for the more complex case with bounded lifespan, when the intruder cannot reuse old commands from the controller.
ATTACKS IN THE PRESENCE OF BOUNDED LIFESPAN AND DELAYS. When a single tank is present, all that the intruder can do is to take the level to any desired value

by cutting communication when the currently issued command sets the output toward its target (though, he is not able to preserve the level there). The case of two tanks is more interesting. This significantly increases the power of the intruder if he can also change the destination of the messages, etc. Assume that the reference level of the two tanks is different, e.g., low^+ vs. $high^+$. Due to the on/off nature of the controller, the level of fluid in the tank will oscillate around the reference level (this is consistent with the transition model described here). In this case the attack trace shows that the intruder can perform a λ-step subversion for any level if for one tank the adversary target level is above the reference while for the other the level is below the reference. This is achievable since the intruder can subvert the command for one of the tanks and send it to the other and additionally it can add delays at will. We present the attack traces, found by the CL-Atse [15] model checker which is part of the AVANTSSAR toolset (www.avantssar.eu), for two distinct cases as outlined below.

Same-phase oscillation. The controller preserves the desired level and the tanks work synchronously, oscillating between $\{low^+, low^-\}$ and $\{high^+, high^-\}$ respectively. The initial states are low^+ for the first and $high^+$ for the second and the initial commands are both negative leading to a first transition to low^- and $high^-$ respectively. The adversary goal is set to $\{med^-, med^+\}$. The attack trace in Figure 2 has 7 steps which use several intruder abilities: *delays* (step i) in which the first tank preserves the previous command), *redirection* (step ii) in which the command for each of the tank is sent to the other) and *replay* (steps iii) to vii) when the command to one of the tanks is also sent to the other; this still respects uniqueness since the command is still freshly issued). In Figure 2, the arrows from the controller to the tank denote the command that is taken as input (or preserved in case when a delay is introduced).

Opposite oscillation. We set the same reference levels ($\{low^+, low^-\}$ and $\{high^+, high^-\}$) and the same attack goals ($\{med^-, med^+\}$). The initial states are low^- and $high^+$ while the initial commands $^+$ and $^-$ respectively that lead to a first transition to low^+ and $high^-$. In this case the two tanks are oscillating in opposite directions at the desired levels. Again 7 attack steps are needed, but the intruder actions are different. Interestingly, in steps i), iv) and vii) the regular commands of the controller are used. In steps ii) and vi) a *redirection* is done, while in step iii) a *delay* is added to both receivers.

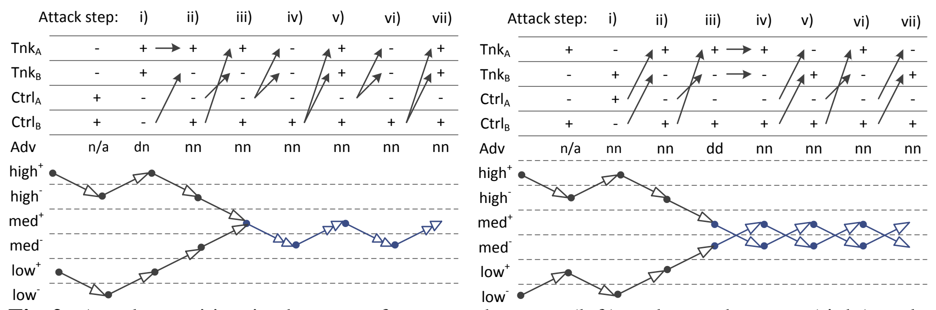
Attack step: i) ii) iii) iv) v) vi) vii)
Tnk_A
Tnk_B
Ctrl_A
Ctrl_B
Adv n/a dn nn nn nn nn nn nn
high+
high-
med+
med-
low+
low-
Attack step: i) ii) iii) iv) v) vi) vii)
Adv n/a nn nn dd nn nn nn nn

Fig. 2: Attack transition in the case of two synchronous (left) and asynchronous (right) tanks (based on the attack trace produced by CL-Atse)

4 Conclusion and Future Work

Bridging Dolev-Yao adversaries and control systems is an interesting and previously largely unaddressed topic. Freshness, ordering and bounded lifespan provide interesting goals for modelling system behaviour. We have shown that neglecting these objectives can lead an adversary to compromise the output of a control system by subverting it to desired target states at will. To the best of our knowledge, this work gives the first account of using formal verification tools to model time-sensitive channel goals in relation to control systems. We are concerned with making the first foundational steps for such analyses. There are several communication protocols widely employed in industrial systems; their investigation is out of scope here, but we see it as relevant future work. The adversary embedded in our practical examples can delay, delete or redirect messages, but he is also able to perform cryptographic operations if these are present in the protocol. Indeed, such operations are quite natural capabilities with the tools employed here. This opens the road to study a broader class of practical examples of more complex systems in the context of Dolev-Yao adversaries.

References

1. C. Alcaraz, R. Roman, P. Najera, and J. Lopez. Security of industrial sensor network-based remote substations in the context of the internet of things. *Ad Hoc Networks*, 11(3):1091 – 1104, 2013
2. M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *Proc. Royal Society of London. Series A, Mathematical and Physical Sciences*, 426(1871):233–271, 1989
3. A. A. Cárdenas, S. Amin, and S. Sastry. Research challenges for the security of control systems. In *Proc. of the 3rd conference on Hot topics in security*, pages 1–6. USENIX, 2008
4. M. Cheminod, A. Pironti, and R. Sisto. Formal vulnerability analysis of a security system for remote fieldbus access. *IEEE Transactions on Industrial Informatics*, 7(1):30–40, 2011
5. G. Delzanno and P. Ganty. Automatic verification of time sensitive cryptographic protocols. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 342–356, 2004
6. J. Edmonds, M. Papa, and S. Shenoi. Security analysis of multilayer SCADA protocols. *Critical Infrastructure Protection*, pages 205–221, 2007
7. A. Girard and G. Pappas. Approximation metrics for discrete and continuous systems. *IEEE Transactions on Automatic Control*, 52(5):782–798, 2007
8. A. Girard and G. Pappas. Approximate bisimulation: A bridge between computer science and control theory. *European Journal of Control*, 17(5):568, 2011
9. G. Lowe. Casper: A compiler for the analysis of security protocols. In *10th Computer Security Foundations Workshop*, pages 18–30. IEEE, 1997
10. G. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, 2003
11. M. Roberts. *Fundamentals of signals and systems*. McGraw-Hill, 2007
12. P. Syverson. A taxonomy of replay attacks. In *7th Computer Security Foundations Workshop*, pages 187–191. IEEE, 1994
13. Y. Tazaki and J. Imura. Discrete-state abstractions of nonlinear systems using multi-resolution quantizer. *Hybrid Systems: Computation and Control*, pages 351–365, 2009
14. A. Teixeira, D. Pérez, H. Sandberg, and K. H. Johansson. Attack models and scenarios for networked control systems. In *Proc. of the 1st conference on High Confidence Networked Systems*, HiCoNS '12, pages 55–64. ACM, 2012
15. M. Turuani. The CL-Atse protocol analyser. In *Proc. of the 17th Int'l. Conference on Term Rewriting and Applications*, LNCS vol. 4098, pages 277–286. Springer, 2006